Gaussian Processes in Response Surface Modeling

Laura P. Swiler
Optimization and Uncertainty Estimation Dept.
Sandia National Laboratories, PO Box 5800
Albuquerque, NM 87185-0370
USA
lpswile@sandia.gov

ABSTRACT

Gaussian processes are used as emulators for expensive computer simulations. Recently, Gaussian processes have also been used to model the "error field" or "code discrepancy" between a computer simulation code and experimental data, and the delta term between two levels of computer simulation (multi-fidelity codes). This work presents the use of Gaussian process models to approximate error or delta fields, and examines how one calculates the parameters governing the process. In multi-fidelity modeling, the delta term is used to correct a lower fidelity model to match or approximate a higher fidelity model. The terms governing the Gaussian process (e.g., the parameters of the covariance matrix) are updated using a Bayesian approach. We have found that use of Gaussian process models requires a good understanding of the method itself and an understanding of the problem in enough detail to identify reasonable covariance parameters. The methods are not "black-box" methods that can be used without some statistical understanding. However, Gaussian processes offer the ability to account for uncertainties in prediction. This approach can help reduce the number of high-fidelity function evaluations necessary in multi-fidelity optimization.

Keywords: Gaussian processes, Multi-Fidelity modeling, Bayesian methods

1. Introduction

Gaussian Process models are used in response surface modeling, especially response surfaces which "emulate" complex computer codes. Gaussian processes have also been widely used for estimation and prediction in geostatistics and similar spatial statistics applications [2]. Good overviews of Gaussian processes are provided in [4], [14] and [15].

A Gaussian process (GP) is defined as follows [15]: A stochastic process is a collection of random variables $\{Y(x) \mid x \in X\}$ indexed by a set X (in most cases, X is $\Re^d$, where d is the number of inputs). The stochastic process is defined by giving the joint probability distribution for every finite subset of variables $Y(x_1), ..Y(x_k)$. A Gaussian process is a stochastic process for which any finite set of Y-variables has a joint multivariate Gaussian distribution. A GP is fully specified by its mean function $\mu(x) = E[Y(x)]$ and its covariance function $C(x, x')$. The basic steps in using a GP are:

1. Define the mean function. The mean function can be any type of function. Often the mean is taken to be zero, but this is not necessary. A common representation, for example in a regression model, is that $y(x) = \sum_j w_j\phi_j(x) = w^T\phi(x)$, where $\{\phi_j\}$ is a set of fixed basis functions and w is a vector of weights. Combining Gaussian process and a Bayesian approach, one places a prior probability distribution over possible function parameters and lets the observed data transform the prior into a posterior.

2. Define the covariance. There are many different types of covariance functions that can be used. At this stage, we shall focus on stationary covariance functions where C(x, x′) is a function of x-x′ and is invariant to shifts of the origin in the input space. A commonly-used covariance function is:

$$C(\mathbf{x},\mathbf{x}') = v_o \exp\{-\sum_{u=1}^{d} \rho_u^2 (\mathbf{x}_u - \mathbf{x}'_u)^2\} \quad (1)$$

   This covariance function involves the product of d squared-exponential covariance functions with different lengthscales on each dimension. The form of this covariance function captures the idea that nearby inputs have highly correlated outputs.

3. Perform the "prediction" calculations. Given a set of n input data points $\{x_1, x_2, .. x_n\}$ and a set of associated observed responses or "targets" $\{z_1, z_2, .. z_n\}$, we use the GP to predict the target z_{n+1} at a new input point x_{n+1}. The target is usually represented as the sum of the "true" response, y, plus an error term: $z_i = y_i + \varepsilon_i$, where ε_i is a zero mean Gaussian random variable with constant variance σ^2_ε. If C is the n×n covariance matrix with entries $C(x_i, x_j)$, then the prior distribution on the targets z_i is N(0,C). The distribution of the predicted term z_{n+1} is conditional on the data $\{z_1, z_2, .. z_n\}$. It is Gaussian with the following mean and variance:

$$E[z_{n+1} \mid z_1, z_2, .. z_n] = k^T C^{-1} z \quad (2)$$

$$Var[z_{n+1} \mid z_1,\ldots, z_n] = C(x_{n+1}, x_{n+1}) - k^T C^{-1} k \quad (3)$$

   where k is the vector of covariances between the n known targets and the new n+1 data point: $k = (C(x_1, x_{n+1}), \ldots\ldots C(x_n, x_{n+1}))^T$, C is the n * n covariance matrix of the original data, and z is the n×1 vector of target values.

   The equations for the mean and variance of the predictive distribution for z_{n+1} both require the inversion of C, an n×n matrix. In general, this is a $O(n^3)$ operation. Neal [13] and Williams [15] claim that this is feasible on modern computers when n is the order of a few hundred, but that it becomes computationally expensive when n is larger than 1000. Booker [1] suggests ways to have well-conditioned Gaussian process models.

Steps 1-3 give the general framework for defining a Gaussian process and using it for prediction. However, the lengthscale parameters in the covariance matrix must be calculated to perform the prediction in equations 2 and 3. There are two main approaches. One is to use maximum likelihood estimation, where one maximizes the likelihood function. This results in point estimates of the covariance parameters. The other approach is to use Monte Carlo Markov Chain (MCMC) sampling to generate posterior distributions on the hyperparameters which govern the covariance function (and the mean function). The assumption of zero mean GPs is often made, so the Bayesian updating only involves hyperparameters governing the covariance function. Since these may be quite complex, one usually still needs a MCMC sampling method to generate the posterior [5]. For example, Neal [13] assumes the ρ^2 terms in the covariance function are distributed as gamma distributions (which themselves are governed by three parameters), so one needs to calculate/update these three parameters for every ρ^2 term.

In terms of Gaussian process implementation, we examined two existing, public domain codes which have capabilities to perform predictions using Gaussian processes. Both of these codes also have the option of using MCMC to generate the posteriors on the hyperparameters.

The first code is called Netlab, which is a collection of Matlab M-files [11]. In addition to Gaussian processes, this code also has capabilities focused on pattern recognition/classification: it has functions for Principal Component Analysis, K-means clustering, radial basis function networks, etc. We found the Matlab scripts reasonably easy to use. The visualization of the Gaussian process surface was one of the advantages of this code.

The second code is Radford Neal's FBM, Flexible Bayesian Modeling [12]. This code is written in C and command line driven. It has a lot of capabilities: Bayesian regression and classification models based on neural nets or Gaussian processes, MCMC, and clustering methods using mixture models. The documentation is reasonable but somewhat cryptic. Specifically, the formulation of the hyperparameters and the specification of the MCMC are not intuitive. We found it difficult to understand what was driving the output results.

We implemented our own version of a Gaussian process model so that we could fully control the form of the basis and the covariance functions, the parameters governing those functions, and the methods to obtain the parameter estimates (Bayesian vs. maximum likelihood, etc.) Our code allows the user to follow the basic steps in generating a Gaussian process: define and initialize the GP, determine estimates of the parameters, calculate the covariance/inverse covariance matrix, define the set of X values for which you want predictions, do a "forward" propagation given the GP structure and hyperparameters to calculate an estimated Y vector for the X inputs along with prediction intervals.

The remainder of this paper is structured as follows. Section 2 defines a framework for identifying a "code discrepancy" term and modeling with a Gaussian process. Section 3 defines a framework for modeling the "error" term between a high- and low-fidelity code, again modeling this error with a Gaussian process, and using it to generate predictions of the high-fidelity results. Section 4 provides some results for a multi-fidelity model. Conclusions are presented in Section 5.

2. Code or Model Discrepancy

Code or model discrepancy is defined as the difference between experimental data and the computer code or simulation model results. The experimental data is assumed equal to some "true" process, $\zeta(x_i)$, plus some error.

$$\text{Experimental data} = z_i = \zeta(x_i) + e_i = \text{Code Output} + \delta(x_i) + e_i \qquad (4)$$

This idea of a code discrepancy term is largely due to work in model calibration. The problem of model calibration is often formulated as finding the parameters that minimize the squared difference between the model-computed data (the predicted data) and the actual experimental data. This approach does not allow for explicit treatment of uncertainty or error in the model itself: the model is considered the "true" deterministic representation of reality. While this approach does have utility, it is far from an accurate mathematical treatment of the true model calibration problem in which both the computed data and experimental data have error bars. We call this approach Calibration under Uncertainty.

Recent research in the Bayesian statistics community has yielded advances in formal statistical methods that address Calibration under Uncertainty. One approach is that of Kennedy and O'Hagan [10], hereafter referred to as KOH1. They formulate a model for calibration data that includes an experimental error term (similar to standard regression) and a model discrepancy term, with a Gaussian process chosen to model the discrepancy. They then use a Bayesian approach to update the statistical parameters associated with the discrepancy term and with the model parameters. The purpose of updating is generally to reduce uncertainty in the parameters through the application of additional information. Reduced uncertainty increases the predictive content of the calibration, or that is the expectation.

In their model calibration work, KOH1 assume that the calibration inputs are supposed to take fixed but unknown values $\theta = (\theta_{1\ldots}\theta_{q2})$. The output of the computer model when the variable inputs are given values $x = (x_1, x_2, \ldots x_{q1})$ and when the calibration inputs are given values $t = (t_1, t_2, \ldots t_{q2})$ is denoted by $\eta(x,t)$. KOH1 differentiate between the unknown value θ of the calibration inputs which we wish to determine (calibrate) and a known particular set of their values, t, which we set as inputs when running the model. The "true" value of the real process when the variable inputs take value x is given by $\zeta(x)$. The code outputs from N runs of the computer code are represented as $y_j = \eta(x_j,t_j)$. The observed data (consisting of n points, where n < N usually) is denoted as $z = (z_1, z_2, \ldots z_n)^T$. In KOH1's formulation, they represent the relationship between the observations, the true process, and the computer model output by the equation:

$$z_i = \zeta(x_i) + e_i = \rho\, \eta(x_i,t_i) + \delta(x_i) + e_i \qquad (5)$$

where e_i is the observation error for the ith observation, ρ is an unknown regression parameter, and $\delta(x)$ is a model discrepancy or model inadequacy function that is independent of the code output $\eta(x,t)$.

A few comments: this is a highly parameterized model, with both the code output $\eta(x,t)$ and $\delta(x)$ represented as Gaussian processes. The formulation in KOH1 involves a complicated joint density function on both $\eta(x,t)$ and $\delta(x)$ which involves six parameter distributions and is intractable to solve for analytically, even with high-dimensional quadrature. KOH1 address this by fixing many of these parameters and use a two stage process, where they estimate the hyperparameters relating to the covariance matrix for the model term separately and before estimating the hyperparameters relating to the covariance matrix of the discrepancy term.

Higdon et. al [6] have applied the idea of Gaussian process modeling in calibration to a number of engineering design problems. Their work is highly recommended for a better understanding of this subject. Because of space limitations, we do not present results of the Gaussian process approach to modeling code discrepancy here. We have applied this approach to a problem where the discrepancy term was fairly linear in the experimental configuration parameters. The GP approach worked well for modeling the discrepancy term between ONE set of experimental data (at a specific experimental configuration, for example, a fixed length component, fixed thermal conductivity values, etc.) and one set of computational model results, where the computational model had input parameters corresponding to the experimental configuration. However, we had four different experimental configurations. While we could produce a GP model for the discrepancy term for each one of the configuration individually, we tried various aggregation techniques with limited success: we were not able to generalize the GP error term in a general way to predict for new configurations which were not one of the four original experimental configurations.

3. High/low fidelity models

Many computational models of high physical fidelity are very expensive in terms of run time. In these cases, we would like to develop an approach to response surface modeling which allows us to construct a response surface based on some low fidelity function evaluations and update the coefficients governing that response surface with a few high fidelity function evaluations. This approach of correcting a low-fidelity response surface and updating it is used in some trust region approaches [3]. A variation on this approach has been developed by Kennedy and O'Hagan [9], who propose constructing an autoregressive model where a higher-fidelity code output is assumed to be an autoregressive function of the lower fidelity code output. We refer to this work [9] as KOH2. Huang et. al [7,8] have expanded on Kennedy and O'Hagan's approach and we have looked at their implementation in detail. The overall idea for multi-fidelity models using a Bayesian autoregressive approach makes the following assumptions:

- Different levels of the same code are correlated in some way.
- The codes have a degree of smoothness in the sense that output values for similar inputs are reasonably close.
- Prior beliefs for each level of code can be modeled using a Gaussian process.

We choose a notation similar to Huang's. If there are l levels of code, l = 1, …, m, the assumption is that:

$$f_l(x) = f_{l-1}(x) + \delta_l(x) \qquad (6)$$

where $\delta_l(x)$ is independent of $f_1(x), f_2(x), \ldots, f_{l-1}(x)$. This means that every level of code differs by the previous level by some delta function. KOH2 [9] assume a slightly more complex autoregressive function:

$$f_l(x) = \rho_{l-1} f_{l-1}(x) + \delta_l(x) \qquad (7)$$

The delta term $\delta_l(x)$ is meant to model the "systematic error" of a lower-fidelity system, (l–1), as compared to the next higher-fidelity system, l. $\delta_l(x)$ is usually small in scale as compared to $f_l(x)$. In KOH2, both the $\delta_l(x)$ and $f_l(x)$ terms are modeled as Gaussian processes.

The major difference between what we have done and what Huang has done is that we have modeled the mean of the GP with a regression term and he has modeled it as a constant. Another difference is that we estimate the

GP for the lower level model, $f_1(x)$, separately from $\delta_1(x)$. Finally, there is an issue of "matching" the models at the points to construct the $\delta(x)$ term. Both KOH2 and Huang evaluate the model at the same data points (the same x values). We evaluated the low and high fidelity data both at the same points to construct the delta term and at different points.

In kriging models, the true, unknown response is assumed to be the sum of a linear model, a term representing the systematic departure (bias) from the linear model, and noise. A Gaussian process model is closely related to a kriging model. The auto-regressive approaches outlined above basically involve a kriging approach for the delta terms:

$$\delta_l(x) = b_l(x)^T \beta_l + Z_l(x) + \varepsilon_l \qquad (l = 1, 2, \ldots, m) \qquad (8)$$

where b_l and β_l are the basis functions and coefficients, respectively, of the linear model. Z_l is the systematic departure and ε_l is the random error. Z_l, is modeled as a zero-mean stationary Gaussian process. Huang and others often assume a constant term for the basis.

The covariance between two points $x = (x_1, \ldots x_d)$ and $x' = (x'_1, \ldots x'_d)$ for the $\delta_l(x)$ function is:

$$\mathrm{cov}[\delta_l(\mathbf{x}), \delta_l(\mathbf{x}')] = \sigma^2_{Z,l} \exp\left[\sum_{j=1}^{d} -\theta_{l,j}(x_j - x'_j)^2\right] \qquad (9)$$

where $\sigma^2_{Z,l}$ is the variance of the stochastic process, and $\theta_{l,j}$ is a "roughness" parameter associated with the dimension j. A larger $\theta_{l,j}$ implies a higher "activity", or lower spatial correlation, within the dimension j.

In our initial implementation, we first estimated a Gaussian process for $f_1(x)$, then estimated $\delta_1(x)$, then summed the two results to obtain $f_2(x)$: $f_2(x) = f_1(x) + \delta_2(x)$.

4. Results

The computational model of interest is the three-dimensional explicit transient dynamics code Presto. Presto is a Lagrangian Finite Element code developed at Sandia National Laboratories. The application here focuses on mechanical deformation. This example has two levels of fidelity: a low fidelity model with approximately 10K finite elements, and a high fidelity model with approximately 50K elements and more detail in the modeling of internal structural elements.

As part of a preliminary investigation, we performed an orthogonal array (OA) parameter study on the model. This allowed us to identify the important parameters in our model. In the following discussion, x is an eight dimensional input space. We ran both the low and high fidelity models at 13 points in the parameter space. These points are shown in Table 1. The output of the low and high fidelity models is displacement. The displacement predictions from the computational codes are denoted as f_{1TRUE} and f_{2TRUE} to differentiate them from the Gaussian process estimates of the low and high fidelity results, which are $f_1(x)$ and $f_2(x)$, respectively. The code output is shown on Table 1 as well. We normalized the output. You can see that the low fidelity code predictions of displacement are larger than the high fidelity code predictions. It is this difference, the "delta term," that we are trying to estimate with a Gaussian process. Then, we will use the Gaussian process to predict what the delta term will be in the case of 6 new points which have values in the X parameters that are outside the domain given in Table 1.

X1	X2	X3	X4	X5	X6	X7	X8	f1true	f2true
15	5	10	0.375	0.0065	2625	12500	4.6	12.86	10.87
20	10	10	0.75	0.013	2750	12500	4.6	13.98	12.04
15	10	15	0.375	0.013	2750	13000	4.6	13.51	11.58
20	5	15	0.75	0.0065	2750	13000	4.7	15.69	13.73
15	10	10	0.75	0.013	2625	13000	4.7	14.46	12.31
15	5	15	0.375	0.013	2750	12500	4.7	13.66	11.48
15	5	10	0.75	0.0065	2750	13000	4.6	13.38	11.18
20	5	10	0.375	0.013	2625	13000	4.7	16.09	14.04
20	10	10	0.375	0.0065	2750	12500	4.7	15.18	12.85
20	10	15	0.375	0.0065	2625	13000	4.6	15.24	13.37
15	10	15	0.75	0.0065	2625	12500	4.7	13.96	11.72
20	5	15	0.75	0.013	2625	12500	4.6	14.30	12.38
20	10	15	0	0	2500	12000	4.5	13.23	11.43

Table 1. Computational model results, f_{1TRUE} and f_{2TRUE}, as a function of eight input variables.

Figure 1 shows the low and high level model results as a function of the first input variable, X1.

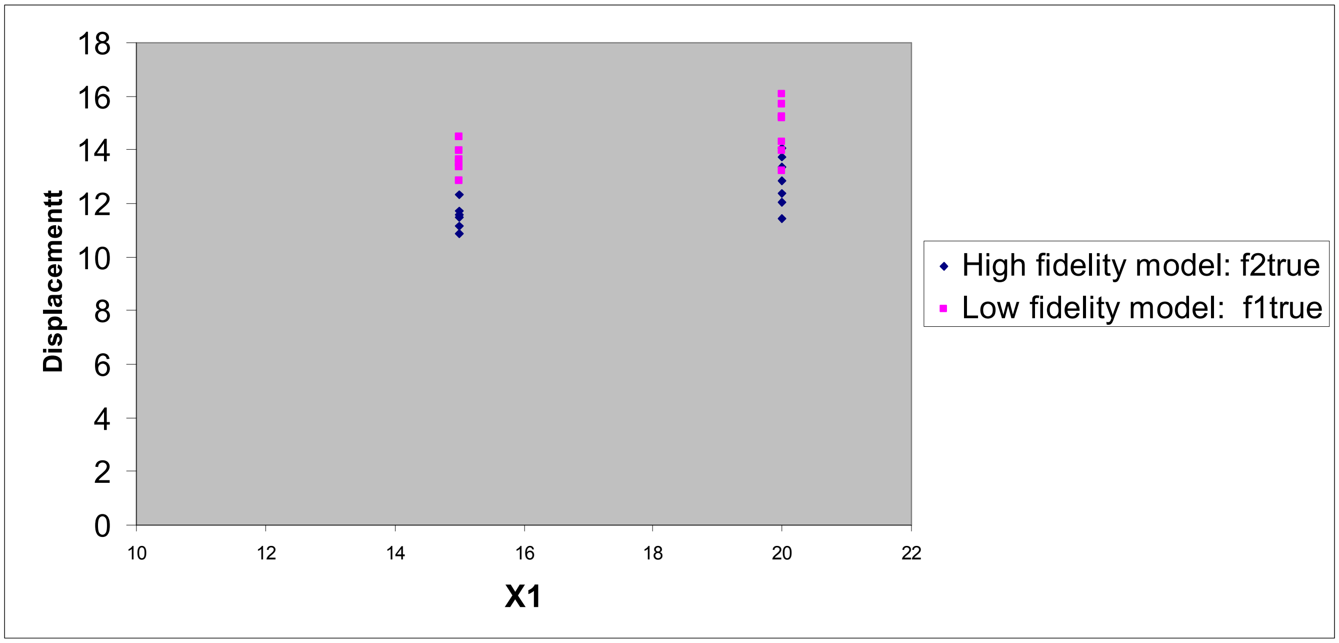

Figure 1. Displacement as a function of X1 for high and low fidelity codes, f_{1TRUE} and f_{2TRUE}.

The first step is to create a Gaussian process estimate of the low fidelity model. Then we create a Gaussian process estimate of the delta term. To obtain the GP estimate $f_1(x)$, we took the 13 points from the low-fidelity parameter study. The low fidelity GP model is: $f_1(x) = b_1(x)^T\beta_1 + Z_1(x) + \varepsilon_1$, where Z_l is modeled as a zero-mean stationary Gaussian process. The coefficients of the regression term are estimated by a standard linear regression procedure, and we used maximum likelihood estimation of the covariance parameters governing the GP term Z_l.

After obtaining $f_1(x)$, we calculated the desired delta function between the high and low-fidelity models as: $f_2(x) - f_1(x) = \delta_2(x)$. That is, we took the actual high-level results from the 13 OA run, subtracted the GP estimate of the function, to obtain the desired values for $\delta_2(x)$. Then, we estimate the GP parameters based on the 13 input points in the table above. In this case, $\delta_2(x)$ is given as: $\delta_2(x) = b_2(x)^T\beta_2 + Z_2(x) + \varepsilon_2$.

With the Gaussian process models of $f_1(x)$ and $\delta_2(x)$ developed, we now can use these to predict $f_2(x)$ at some new points. We chose six new points shown in Table 2. We ran the high fidelity model at these points to check the accuracy of our GP estimate, but we did NOT run the low fidelity model at these points. Instead, we used the GP estimate $f_1(x)$. If the low fidelity function evaluations were cheap enough computationally, one could use the code results for the low fidelity model and not use a GP approximation of the low fidelity model. Note that our approach has two Gaussian process terms added together to get the estimate of the high fidelity model:
$f_2(x) = f_1(x) + \delta_2(x)$. However, in practice, it may be desirable just to create a Gaussian process model for the delta term if the "true" low fidelity calculations are available.

X1	X2	X3	X4	X5	X6	X7	X8	F2true	F2predicted	Error	Relative % Error
17	7	13	0.5	0.01	2700	12500	5	14.70	14.49	0.21	1.46
25	10	10	0.75	0.013	2800	12500	4.6	12.77	13.41	-0.63	4.96
15	10	15	0.9	0.02	2700	12000	4.8	11.18	11.46	-0.28	2.50
20	5	15	0.2	0.005	2800	14000	4.7	15.41	15.40	0.01	0.06
15	10	15	0.2	0.005	2800	14000	4.8	14.89	14.67	0.22	1.46
20	5	15	0.9	0.02	2700	12000	4.7	11.90	12.18	-0.29	2.40

Table 2. New X input points where we compare the GP prediction, $f_{2predicted}$, with the high fidelity model, f_2.

Figure 2 shows the "true" high level results for these six points, the GP predictions of these results, as well as the GP predictions of the delta term and the low level model results.

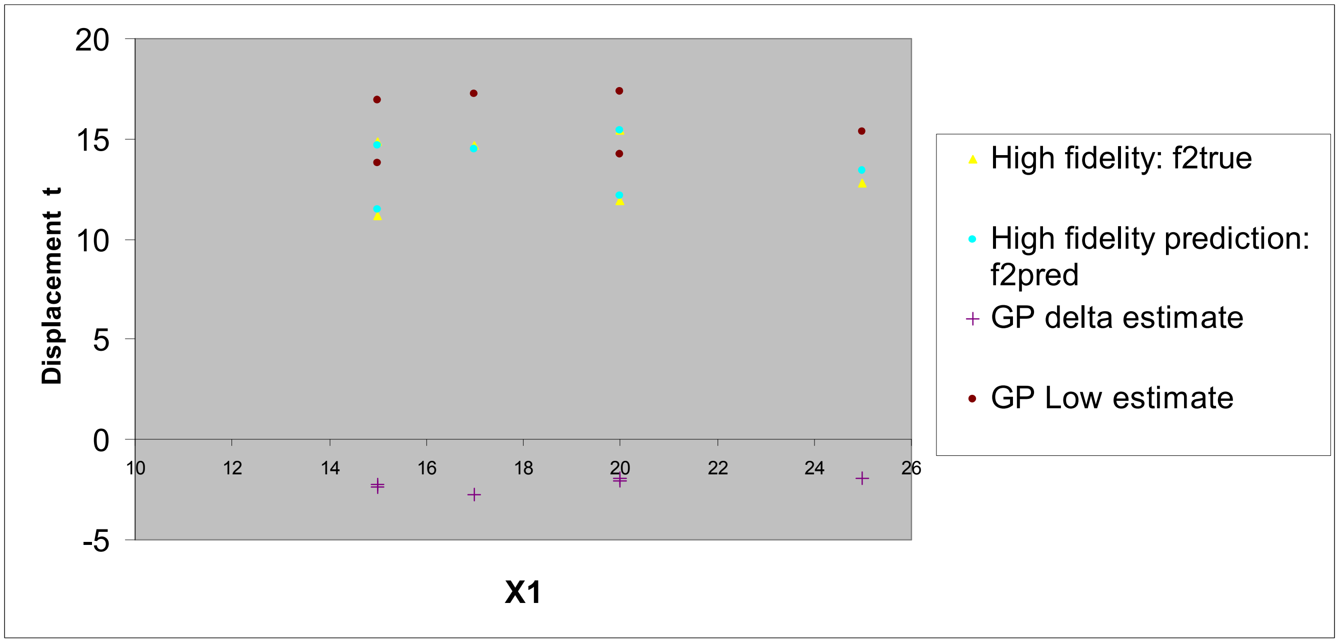

Figure 2. High fidelity prediction, $f_{2predicted}$, compared with the true high fidelity model, f_2.

Overall, we have very good agreement: the displacement predicted by the GP autoregressive model and the displacement obtained by the high fidelity "true" code calculation are very similar. The percentage error in the GP model is less than 5% in all six cases shown in Table 2 and is less than 3% in five of the cases. The largest error, for point 2, is due to the fact that this point represents a significant extrapolation of X1: the data upon which the GP models were built (the 13 points in Table 1) only involved X1 at values of 15 and 20, but this point has X1 at a value of 25. Note that we constructed Gaussian process models for the low fidelity model and for the delta term only based on 13 points in 8 dimensional space. Given that we are using these GP models to predict the output at 6 new points (where each new point involves extrapolation on at least one dimension), the predictions look good. Also note that the prediction of the low fidelity model gives higher estimates of displacement than the high fidelity model and the delta term is always negative. This is what we expect based on the original 13 data points.

Based on these results, we can say that an autoregressive approach based on GP models seems reasonable to pursue. We are currently addressing some issues relating to implementation of the covariance terms in the GP models. Using Gaussian process models for low fidelity results and an estimate of the delta term between high and low fidelity to predict high fidelity results is a promising approach. This could have many applications, especially in optimization and uncertainty quantification problems.

5. Summary

Gaussian processes have been used as emulators for expensive computer simulations: they provide good response surface approximations, especially locally around existing points. The Bayesian statistics community has developed some ideas for calibration and also for multi-fidelity approaches based on Gaussian processes. Specifically, Gaussian processes are used to model the "code discrepancy" or model discrepancy term in calibration (the difference between the computational model results and experimental data). In multi-fidelity modeling, a "delta" term is used to correct a lower fidelity model to match or approximate a higher fidelity model. The delta term is also approximated with a Gaussian process. In both the calibration and multi-fidelity case, the terms governing the Gaussian process (e.g., the parameters of the covariance matrix) can be determined by a maximum likelihood optimization or "updated" using a Bayesian approach. We have found that use of Gaussian process models requires a good understanding of the method itself and an understanding of the problem in enough detail to normalize parameters, identify reasonable covariance parameters, etc. The methods are not "black-box" methods that can be used without some statistical understanding. However, GPs offer the ability to account for uncertainties as well as provide an estimate of a delta or discrepancy term. That is why they are very useful in design problems and why we think they have particular applicability to optimization under uncertainty. Our preliminary research has shown that a multi-fidelity model, where a high fidelity simulation is approximated by a lower fidelity simulation plus a GP delta term, is viable. A GP provides a reasonable functional form for the delta term; a GP approach can greatly help reduce the number of high-fidelity function evaluations necessary; and the GPs have good prediction capabilities. For these reasons, we are excited about using the Gaussian process approach for design optimization problems under uncertainty.

6. References

[1] Booker, A. "Well-conditioned Kriging Models for Optimization of Computer Simulations." *Technical Document Series, M&CT-TECH-002,* Phantom Works, Mathematics and Computing Technology, The Boeing Company, Seattle, WA, 2000.

[2] Cressie, N. A. C. *Statistics for Spatial Data*, Wiley, New York, 1993.

[3] Eldred, M.S., Giunta, A.A., and Collis, S.S, "Second-Order Corrections for Surrogate-Based Optimization with Model Hierarchies," paper AIAA-2004-4457 in *Proceedings of the 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Albany, NY, Aug. 30 - Sept. 1, 2004.

[4] Gibbs, M. and D. J. C. MacKay. *Efficient Implementation of Gaussian Processes*. On the Gaussian process web site: http://www.cs.toronto.edu/~carl/gp.html.

[5] Gilks, W.R., S. Richardson, and D.J. Spiegelhalter (1996). *Markov Chain Monte Carlo in Practice*. Chapman and Hall/CRC, Boca Raton, 1996.

[6] Higdon, D., Kennedy, M., Cavendish, J., Cafeo, J., and R. D. Ryne. "Combining Field Data and Computer Simulations for Calibration and Prediction." *SIAM Journal on Scientific Computing.* **26**, pp. 448- 466, 2004.

[7] Huang, D., T.T. Allen, W. I. Notz, and R. A. Miller, "Sequential Kriging Optimization Using Multiple Fidelity Evaluations", submitted to *Structural and Multidisciplinary Optimization*.

[8] Huang, D., Allen, T. T., Notz, W. I., and Zheng, N., "Global Optimization of Stochastic Black-Box Systems via Sequential Kriging Meta-Models", accepted by the *Journal of Global Optimization*.

[9] Kennedy, M. C. and A. O'Hagan. "Predicting the output from a complex computer code when fast approximations are available." *Biometrika*, **87**, pp. 1-13, 2000.

[10] Kennedy, M. C. and A. O'Hagan. "Bayesian Calibration of Computer Models." *Journal of the Royal Statistical Society*, **63**, pp. 425-464, 2001.

[11] Nabney, I. and C. Bishop. Netlab software. Documentation and software at: www.ncrg.aston.ac.uk/**netlab**/

[12] Neal, R. Flexible Bayesian Software documentation:
http://www.cs.toronto.edu/~radford/fbm.software.html

[13] Neal, R. *Monte Carlo Implementation of Gaussian Process Models for Bayesian Regression and Classification*. Technical Report 9702, Dept. of Statistics, University of Toronto, 1997.

[14] Rasmussen, Carl. *Evaluation of Gaussian Processes and Other Methods for Nonlinear Regression.* Ph.D. Thesis, University of Toronto, 1996.

[15] Williams, C. "Gaussian Processes" chapter in *The Handbook of Brain Theory and Neural Networks*, M. Arbib, ed. Cambridge, MA: MIT Press, 2002.